On Variants of the Matroid Secretary Problem

Shayan Oveis Gharan* Jan Vondrák†

Abstract

We present a number of positive and negative results for variants of the matroid secretary problem. Most notably, we design a constant-factor competitive algorithm for the "random assignment" model where the weights are assigned randomly to the elements of a matroid, and then the elements arrive on-line in an adversarial order (extending a result of Soto [21]). This is under the assumption that the matroid is known in advance. If the matroid is unknown in advance, we present an $O(\log r \log n)$-approximation, and prove that a better than $O(\log n/\log\log n)$ approximation is impossible. This resolves an open question posed by Babaioff et al. [3].

As a natural special case, we also consider the classical secretary problem where the number of candidates n is unknown in advance. If n is chosen by an adversary from $\{1,\ldots,N\}$, we provide a nearly tight answer, by providing an algorithm that chooses the best candidate with probability at least $1/(H_{N-1}+1)$ and prove that a probability better than $1/H_N$ cannot be achieved (where H_N is the N-th harmonic number).

1 Introduction

The secretary problem is a classical problem in probability theory, with obscure origins in the 1950's and early 60's ([12, 18, 9]; see also [11]). The goal in this problem is to select the best candidate out of a sequence revealed one-by-one, where the ranking is uniformly random. A classical solution finds the best candidate with probability at least $1/e$ [11]. Over the years a number of variants have been studied, starting with [13] where multiple choices and various measures of success were considered for the first time.

Recent interest in variants of the secretary problem has been motivated by applications in on-line mechanism design [15, 19, 3], where items are being sold to agents arriving on-line, and there are certain constraints on which agents can be simultaneously satisfied. Equivalently, one can consider a setting where we want to hire several candidates under certain constraints. Babaioff, Immorlica and Kleinberg [3] formalized the *matroid secretary problem* and presented constant-factor competitive algorithms for several interesting cases. The general problem formulated in [3] is the following.

Matroid secretary problem. Given a matroid $\mathcal{M}=(E,\mathcal{I})$ with non-negative weights assigned to E; the only information known up-front is the number of elements $n:=|E|$. The elements of E arrive in a random order, with their weights revealed as they arrive. When an element arrives, it can be selected or rejected. The selected elements must always form an independent set in $\mathcal{M}$, and

*Stanford University, Stanford, CA; shayan@stanford.edu; this work was done while the author was at IBM Almaden Research Center, San Jose, CA.

†IBM Almaden Research Center, San Jose, CA; jvondrak@us.ibm.com

a rejected element cannot be considered again. The goal is to maximize the expected weight of the selected elements.

Additional variants of the matroid secretary problem have been proposed and studied, depending on how the input ordering is generated, how the weights are assigned and what is known in advance. In all variants, elements with their weights arrive in an on-line fashion and an algorithm must decide irrevocably whether to accept or reject an element once it has arrived. We attempt to bring some order to the multitude of models and we classify the various proposed variants as follows.

Ordering of matroid elements on the input:

- AO = Adversarial Order: the ordering of elements of the matroid on the input is chosen by an adversary.
- RO = Random Order: the elements of the matroid arrive in a random order.

Assignment of weights:

- AA = Adversarial Assignment: weights are assigned to elements of the matroid by an adversary.
- RA = Random Assignment: the weights are assigned to elements by a random permutation of an adversarial set of weights (independent of the input order, if that is also random).

Prior information:

- MK = Matroid Known: the matroid is known beforehand (by means of an independence oracle).
- MN = Matroid - n known: the matroid is unknown but the cardinality of the ground set is known beforehand.
- MU = Matroid - Unknown: nothing about the matroid is known in advance; only subsets of the elements that arrived already can be queried for independence.

For example, the original variant of the matroid secretary problem [3], where the only information known beforehand is the total number of elements, can be described as RO-AA-MN in this classification. We view this as the primary variant of the matroid secretary problem.

We also consider variants of the classical secretary problem; here, only 1 element should be chosen and the goal is to maximize the probability of selecting the best element.

Classical secretary problems:

- CK = Classical - Known n: the classical secretary problem where the number of elements in known in advance.
- CN = Classical - known upper bound N: the classical secretary problem where the number of elements is chosen adversarially from $\{1, \ldots, N\}$, and N is known in advance.
- CU = Classical - Unknown n: the classical secretary problem where no information on the number of elements is known in advance.

Since the independent sets of the underlying matroid in this model are independent of the particular labeling of the ground set (i.e., RO-AA-CK, AO-RA-CK and RA-RO-CK models are equivalent), we just use the weight assignment function to characterize different variants of this model. The classical variant of the secretary problem which allows a $1/e$-approximation would be described as RA-CK. The variant where the number of elements n is not known in advance is very natural — and has been considered under different stochastic models where n is drawn from a particular

distribution [23, 1] — but the worst-case scenario does not seem to have received attention. We denote this model RA-CU, or RA-CN if an upper bound on the number of candidates is given. In the model where the input ordering of weights is adversarial (AA-CK), it is easy to see that no algorithm achieves probability better than $1/n$ [5]. We remark that variants of the secretary problem with other objective functions have been also proposed, such as discounted profits [2], and submodular objective functions [4, 14]. We do not discuss these variants here.

1.1 Recent related work

The primary variant of matroid secretary problem (RO-AA-MN model) was introduced in [3]. In the following, let n denote the total number of elements and r the rank of the matroid. An $O(\log r)$-approximation for the RO-AA-MN model was given in [3]. It was also conjectured that a constant-factor approximation should exist for this problem and this question is still open. Very recently, Chakraborty and Lachish [7] improved [3] by giving an $O(\sqrt{\log r})$-approximation algorithm. Constant-factor approximations were given in [3] for some special cases such as partition matroids and graphic matroids with a given explicit representation. Further, constant-factor approximations were given for transversal matroids [8, 20] and laminar matroids [17]. However, even for graphic matroids in the RO-AA-MK model when the graphic matroid is given by an oracle, no constant factor is known.

Babaioff et al. in [3] also posed as an open problem whether there is a constant-factor approximation algorithm for the following two models: Assume that a set of n numerical values are assigned to the matroid elements using a random one-to-one correspondence but that the elements are presented in an adversarial order (AO-RA in our notation). Or, assume that both the assignment of values and the ordering of the elements in the input are random (RO-RA in our notation). The issue of whether the matroid is known beforehand is left somewhat ambiguous in [3].

In a recent work [21], José Soto partially answered the second question, by designing a constant-factor approximation algorithm in the RO-RA-MK model: An adversary chooses a list of non-negative weights, which are then assigned to the elements using a random permutation, which is independent of the random order at which the elements are revealed. The matroid is known in advance here.

1.2 Our results

Matroid secretary. We resolve the question from [3] concerning adversarial order and random assignment, by providing a constant-factor approximation algorithm in the AO-RA-MK model, and showing that no constant-factor approximation exists in the AO-RA-MN model. More precisely, we prove that there is a $40/(1-1/e)$-approximation in the AO-RA-MK model, i.e. in the model where weights are assigned to the elements of a matroid randomly, the elements arrive in an adversarial order, and the matroid is known in advance. We provide a simple thresholding algorithm, which gives a constant-factor approximation for the AO-RA-MK model when the matroid $\mathcal{M}$ is uniformly dense. Then we use the principal sequence of a matroid to design a constant-factor approximation for any matroid using the machinery developed by Soto [21]. (Subsequently to our work, Soto [22] improved our approximation factor in the AO-RA-MK model to $16/(1-1/e)$.)

On the other hand, if the matroid is not known in advance (AO-RA-MN model), we prove that the problem cannot be approximated better than within $\Omega(\log n/\log\log n)$. This holds even in the special case of rank 1 matroids; see below. On the positive side, we show an $O(\log r \log n)$-

approximation for this model. We achieve this by providing an $O(\log r)$-approximation thresholding algorithm for the AO-AA-MU model (when both the input ordering and the assignment of weights to the elements the matroid are adversarial), when an estimate on the weight of the largest non-loop element is given. Here, the novel technique is to employ a dynamic threshold depending on the rank of the elements seen so far.

Classical secretary with unknown n. A very natural question that arises in this context is the following. Consider the classical secretary problem, where we want to select 1 candidate out of n. The classical solution relies on the fact that n is known in advance. However, what if we do not know n in advance, which would be the case in many practical situations? We show that if an upper bound N on the possible number of candidates n is given (RA-CN model: i.e., n is chosen by an adversary from $\{1, \dots, N\}$), the best candidate can be found with probability $1/(H_{N-1}+1)$, while there is no algorithm which achieves probability better than $1/H_N$ (where $H_N = \sum_{i=1}^{N} \frac{1}{i}$ is the N-th harmonic number).

In the model where we maximize the expected value of the selected candidate, and n is chosen adversarially from $\{1, \dots, N\}$, we prove we cannot achieve approximation better than $\Omega(\log N / \log \log N)$. On the positive side, even if no upper bound on n is given, the maximum-weight element can be found with probability $\epsilon / \log^{1+\epsilon} n$ for any fixed $\epsilon > 0$. We remark that similar results follow from [16] and [10] where an equivalent problem was considered in the context of online auctions. More generally, for the matroid secretary problem where no information at all is given in advance (RO-AA-MU), we achieve an $O(\frac{1}{\epsilon} \log r \log^{1+\epsilon} n)$ approximation for any $\epsilon > 0$. See Table 1 for an overview of our results.

Problem	New approximation	New hardness
RA-CN	$H_{N-1}+1$	H_N
RA-CU	$O(\frac{1}{\epsilon} \log^{1+\epsilon} n)$	$\Omega(\log n)$
AO-RA-MK	$40/(1-1/e)$	-
AO-RA-MN	$O(\log r \log n)$	$\Omega(\log n / \log \log n)$
AO-RA-MU	$O(\frac{1}{\epsilon} \log r \log^{1+\epsilon} n)$	$\Omega(\log n / \log \log n)$
RO-AA-MU	$O(\frac{1}{\epsilon} \log r \log^{1+\epsilon} n)$	$\Omega(\log n / \log \log n)$

Table 1: Summary of results

Organization. In section 2 we provide a $40/(1-1/e)$ approximation algorithm for the AO-RA-MK model. In section 3 we provide an $O(\log n \log r)$ approximation algorithm for the AO-RA-MN model, and an $O(\frac{1}{\epsilon} \log r \log^{1+\epsilon} n)$ approximation for the RO-AA-MU model. Finally, in section 4 we provide a $(H_{N-1}+1)$-approximation and H_N-hardness for the RA-CN model.

2 Approximation for adversarial order and random assignment

In this section, we derive a constant-factor approximation algorithm for the AO-RA-MK model, i.e. assuming that the ordering of the elements of the matroid is adversarial but weights are assigned to the elements by a random permutation, and the matroid is known in advance. We build on Soto's algorithm [21], in particular on his use of the *principal sequence of a matroid* which effectively

reduces the problem to the case of a uniformly-dense matroid while losing only a constant factor $(1 - 1/e)$. Interestingly, his reduction only requires the randomness in the assignment of weights to the elements but not a random ordering of the matroid on the input. Hence, it is sufficient to obtain a constant factor for uniformly dense matroids. Recall that the density of a set in a matroid $\mathcal{M} = (E, \mathcal{I})$ is the quantity $\gamma(S) = \frac{|S|}{rank(S)}$. A matroid is uniformly dense, if $\gamma(S) \leq \gamma(E)$ for all $S \subseteq E$. We present a simple thresholding algorithm which works in the AO-RA-MK model (i.e. even for an adversarial ordering of the elements) for any uniformly dense matroid. Combining our algorithm with Sotos reduction [21, Lemma 4.4], we obtain a constant-factor approximation algorithm for the matroid secretary problem in AO-RA-MK model.

Throughout this section we use the following notation. Let $\mathcal{M} = (E, I)$ be a uniformly dense matroid of rank r. This also means that $\mathcal{M}$ contains no loops. Let $|E| = n$ and let $e_1, e_2, \ldots, e_n$ denote the ordering of the elements on the input, which is chosen by an adversary (i.e. we consider the worst case). Furthermore, the adversary also chooses $W = \{w_1 > w_2 > \ldots > w_n\}$, a set of non-negative weights. The weights are assigned to the elements of $\mathcal{M}$ via a random bijection $\omega : E \to W$. For a weight assignment ω, we denote by $w(S) = \sum_{e \in S} \omega(e)$ the weight of a set S, and by $\omega(S) = \{\omega(e) : e \in S\}$ the set of weights assigned to S. We also let $\mathrm{OPT}(\omega)$ be the maximum-weight independent set in $\mathcal{M}$.

2.1 Approximation for uniformly dense matroids

We show that there is a simple thresholding algorithm which includes each of the topmost $\lfloor r/4 \rfloor$ weights (i.e. $w_1, \ldots, w_{\lfloor r/4 \rfloor}$) with a constant probability. This will give us a constant factor approximation algorithm, as $w(\mathrm{OPT}(\omega)) \leq \sum_{i=1}^{r} w_i$, where $w_1 > w_2 > \ldots > w_r$ are the r largest weights in W. It is actually important that we compare our algorithm to the quantity $\sum_{i=1}^{r} w_i$, because this is needed in the reduction to the uniformly dense case.

The main idea is that the randomization of the weight assignment makes it very likely that the optimum solution contains many of the top weights in W. Therefore, instead of trying to compute the optimal solution with respect to ω, we can just focus on catching a constant fraction of the top weights in W. Let $A = \{e_1, \ldots, e_{n/2}\}$ denote the first half of the input and $B = \{e_{n/2+1}, \ldots, e_n\}$ the second half of the input. Note that the partition into A and B is determined by the adversary and not random. Our solution is to use the $\lfloor r/4 \rfloor + 1$-st topmost weight in the "sampling stage" A as a threshold and then include every element in B that is above the threshold and independent of the previously selected elements. Details are described in Algorithm 1.

Theorem 2.1. *Let $\mathcal{M}$ be a uniformly dense matroid of rank r, and $\mathrm{ALG}(\omega)$ be the set returned by Algorithm 1 when the weights are defined by a uniformly random bijection $\omega : E \to W$. Then*

$$\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) \right] \geq \frac{1}{40} \sum_{i=1}^{r} w_i$$

where $\{w_1 > w_2 > \ldots > w_r\}$ are the r largest weights in W.

If $r < 12$, the algorithm finds and returns the largest weight w_1 with probability $1/e$ (step 2; the optimal algorithm for the classical secretary problem). Therefore, for $r < 12$, we have $\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) \right] \geq \frac{1}{11e} \sum_{i=1}^{r} w_i > \frac{1}{40} \sum_{i=1}^{r} w_i$.

Algorithm 1 Thresholding algorithm for uniformly dense matroids in AO-RA-MK model

Input: A uniformly dense matroid $\mathcal{M} = (E, \mathcal{I})$ of rank r.
Output: An independent set $\mathrm{ALG} \subseteq E$.

if $r < 12$ **then**
  run the optimal algorithm for the classical secretary problem, and return the resulting singleton.
end if
$\mathrm{ALG} \leftarrow \emptyset$
Observe a half of the input (elements of A) and let w^* be the $(\lfloor r/4 \rfloor + 1)^{st}$ largest weight among them.
for each element $e \in B$ arriving afterwards **do**
  if $\omega(e) > w^*$ and $ALG \cup \{e\}$ is independent **then**
    $\mathrm{ALG} \leftarrow \mathrm{ALG} \cup \{e\}$
  end if
end for
return ALG

For $r \geq 12$, we prove that each of the topmost $\lfloor r/4 \rfloor$ weights will be included in $\mathrm{ALG}(\omega)$ with probability at least $1/8$. Hence, we will obtain

$$\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) \right] \geq \frac{1}{8} \sum_{i=1}^{\lfloor r/4 \rfloor} w_i \geq \frac{1}{40} \sum_{i=1}^{r} w_i. \tag{1}$$

Let $t = 2\lfloor r/4 \rfloor + 2$. Define $C'(\omega) = \{e_j : \omega(e_j) \geq w_t\}$ to be the set of elements of $\mathcal{M}$ which get one of the top t weights. Also let $A'(\omega) = C'(\omega) \cap A$ and $B'(\omega) = C'(\omega) \cap B$. Moreover, for each $1 \leq i \leq t$ we define $C'_i(\omega) = \{e_j : \omega(e_j) \geq w_t \ \& \ \omega(e_j) \neq w_i\}$, $A'_i(\omega) = C'_i(\omega) \cap A$ and $B'_i(\omega) = C'_i(\omega) \cap B$, i.e. the same sets with the element of weight w_i removed.

First, we fix $i \leq \lfloor r/4 \rfloor$ and argue that the size of $B'_i(\omega)$ is smaller than $A'_i(\omega)$ with probability $1/2$. Then we will use the uniformly dense property of $\mathcal{M}$ to show that the span of $B'_i(\omega)$ is also quite small with probability $1/2$ and consequently w_i has a good chance of being included in $\mathrm{ALG}(\omega)$.

Claim 2.2. *Let $\mathcal{M}$ be a uniformly dense matroid of rank r, $t = 2\lfloor r/4 \rfloor + 2$, $1 \leq i \leq \lfloor r/4 \rfloor$, and $B'_i(\omega)$ defined as above. Then we have*

$$\mathbf{P}_\omega \left[|B'_i(\omega)| \leq \lfloor r/4 \rfloor \right] = 1/2. \tag{2}$$

Proof. Consider $C'_i(\omega)$, the set of elements receiving the top t weights except for w_i. This is a uniformly random set of odd size $t - 1 = 2\lfloor r/4 \rfloor + 1$. By symmetry, with probability exactly $1/2$, a majority of these elements are in A, and hence at most $\lfloor r/4 \rfloor$ of these elements are in B, i.e. $|B'_i(\omega)| \leq \lfloor r/4 \rfloor$. □

Now we consider the element receiving weight w_i. We claim that this element will be included in $\mathrm{ALG}(\omega)$ with a constant probability.

Claim 2.3. *Let $\mathcal{M}$ be a uniformly dense matroid of rank r, and $i \leq \lfloor r/4 \rfloor$. Then*

$$\mathbf{P}_\omega \left[\omega^{-1}(w_i) \in \mathrm{ALG}(\omega) \right] \geq 1/8.$$

Proof. Condition on $C'_i(\omega) = S$ for some particular set S of size $t-1$ such that $|B'_i(\omega)| = |S \cap B| \leq \lfloor r/4 \rfloor$. This fixes the assignment of the top t weights except for w_i. Under this conditioning, weight w_i is still assigned uniformly to one of the remaining $n - t + 1$ elements.

Since we have $|A'_i(\omega)| = |S \cap A| \geq \lfloor r/4 \rfloor + 1$, the threshold w^* in this case is one of the top t weights and the algorithm will never include any weight outside of the top t. Therefore, we have $\text{ALG}(\omega) \subseteq B'(\omega)$. The weight w_i is certainly above w^* because it is one of the top $\lfloor r/4 \rfloor$ weights. It will be added to $\text{ALG}(\omega)$ whenever it appears in B and it is not in the span of previously selected elements. Since all the previously included elements must be in $B'_i(\omega) = S \cap B$, it is sufficient to avoid being in the span of $S \cap B$. To summarize, we have

$$\omega^{-1}(w_i) \in B \setminus span(S \cap B) \quad \Rightarrow \quad \omega^{-1}(w_i) \in \text{ALG}(\omega).$$

What is the probability that this happens? Similar to the proof of [21, Lemma 3.1], since $\mathcal{M}$ is uniformly dense, we have

$$\frac{|span(S \cap B)|}{|S \cap B|} \leq \frac{|span(S \cap B)|}{rank(span(S \cap B))} \leq \frac{n}{r} \Longrightarrow |span(S \cap B)| \leq \frac{n}{r}|S \cap B| \leq \frac{n}{4}$$

using $|S \cap B| \leq \lfloor r/4 \rfloor$. Therefore, there are at least $n/4$ elements in $B \setminus span(S \cap B)$. Given that the weight w_i is assigned uniformly at random among $n - t$ possible elements, we get

$$\mathbf{P}_\omega \left[\omega^{-1}(w_i) \in B \setminus span(S \cap B) \mid C'_i(\omega) = S\right] \geq \frac{n/4}{n-t} \geq \frac{1}{4}.$$

Since this holds for any S such that $|S \cap B| \leq \lfloor r/4 \rfloor$, and $S \cap B = C'_i \cap B = B'_i(\omega)$, it also holds that

$$\mathbf{P}_\omega \left[\omega^{-1}(w_i) \in B \setminus span(B'_i(\omega)) \mid |B'_i(\omega)| \leq \lfloor r/4 \rfloor\right] \geq \frac{1}{4}.$$

Using Claim 2.2, we get $\mathbf{P}_\omega \left[\omega^{-1}(w_i) \in B \setminus span(B'_i(\omega))\right] \geq 1/8$. □

This finishes the proof of Theorem 2.1.

2.2 Extension to general matroids

In this section we describe the final $40/(1-1/e)$ approximation algorithm for AO-RA-MK model for general matroids. The algorithm is based on Soto's algorithm [21], by decomposing the underlying matroid into a sequence of principal minors and then running Algorithm 1 in parallel on each of them separately.

Algorithm 2 Thresholding algorithm for matroid secretary problem in AO-RA-MK model

Input: A matroid $\mathcal{M} = (E, \mathcal{I})$.

Output: An independent set $\text{ALG} \subseteq E$.

1: Compute the sequence of principal minors $(\mathcal{M}_i)_{i=1}^k$: Initialize $k = 0$. While $\bigcup_{i=1}^k E_i \neq E$, let E_{k+1} be the densest set in the matroid $\mathcal{M}/\bigcup_{i=1}^k E_i$, define $\mathcal{M}_{k+1} = (\mathcal{M}/\bigcup_{i=1}^k E_i)|E_{k+1}$, and increment k.

2: Run Algorithm 1 in parallel on each $\mathcal{M}_i$ to get a solution I_i, and return $ALG = \bigcup_{i=1}^k I_i$.

We use Soto's lemma to argue that if the weights are assigned randomly to the elements, and we achieve an α-fraction of the sum of the r_i topmost weights in each principal minor $\mathcal{M}_i$, then we obtain an $\alpha/(1-1/e)$ approximation overall.

Interestingly, it is necessary to know the matroid in advance, in order to discriminate the dense parts of the matroid from the sparse parts (by computing the principal minors), and try to handle them separately. Otherwise, as we prove later, no algorithm can do better than an $O(\log n/\log\log n)$ approximation.

Corollary 2.4. *Algorithm 2 gives a $\frac{40}{1-1/e}$-approximation in the AO-RA-MK model.*

Proof. Similar to the proof of Theorem 2.1, let $e_1,\ldots,e_n$ be the sequence of elements of M designed by the adversary and $W = w_1 > \ldots > w_n$ be the hidden list of weights. Let $\mathcal{M}_i$, $1 \le i \le k$, be the sequence of principal minors of $\mathcal{M}$, with ground set E_i and rank r_i, and let $\mathcal{P}$ denote a partition matroid as defined in [21, Section 4], with ground set E and independent sets

$$\mathcal{I}(\mathcal{P}) = \left\{ \bigcup_{i=1}^{k} I_i : I_i \subseteq E_i, |I_i| \le r_i \right\}.$$

For a uniformly random bijection $\omega : E \to W$, let $\mathrm{OPT}_{\mathcal{P}}(\omega)$ be the maximum weight of an independent set in matroid $\mathcal{P}$, and $\mathrm{ALG}(\omega)$ be the set returned by Algorithm 2. Conditioning on the set of weights assigned to the elements of each block E_i, the elements in E_i receive a random permutation of this set of weights. Since each $\mathcal{M}_i$ is uniformly dense, By Theorem 2.1, Algorithm 2 recovers in expectation a 1/40-fraction of the sum of the heaviest r_i weights assigned to elements in E_i. However, the union of the heaviest r_i elements in each E_i is indeed the optimum solution in the partition matroid $\mathcal{P}$. By removing the conditioning we get

$$\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))\right] \ge \frac{1}{40}\mathbf{E}_\omega\left[\mathrm{OPT}_{\mathcal{P}}(\omega)\right]. \tag{3}$$

Moreover, Soto in [21] proved that $\mathbf{E}_\omega[\mathrm{OPT}_{\mathcal{P}}]$ is only a constant factor away from the optimum of $\mathbf{E}_\omega[\mathrm{OPT}_{\mathcal{M}}]$.

Lemma 2.5 (Soto [21])**.** $\mathbf{E}_\omega\left[w(\mathrm{OPT}_{\mathcal{P}})(\omega))\right] \ge (1-1/e)\mathbf{E}_\omega\left[w(\mathrm{OPT}_{\mathcal{M}}(\omega))\right].$

This proves the corollary. □

3 Approximation algorithms for unknown matroids

In this section we will be focusing mainly on the AO-RA-MN model. i.e. assuming that the ordering of the elements of the matroid is adversarial, weights are assigned randomly, but the matroid is unknown, and the algorithm only knows n in advance. We present an $O(\log n \log r)$ approximation algorithm for the AO-RA-MN model, where n is the number of elements in the ground set and r is the rank of the matroid. It is worth noting that in these models the adversary may set some of the elements of the matroid to be loops, and the algorithm does not know the number of loops in advance. For example it might be the case that after observing the first 10 elements, the rest are all loops and thus the algorithm should select at least one of the first 10 elements with some non-zero

probability. This is the idea of the counterexample in section 4 (Corollary 4.4), where we reduce AO-RA-MN, AO-RA-MU models to RA-CN, RA-CU models respectively, and thus we show that there is no constant-factor approximation for either of the models. In fact, no algorithm can do better than $\Omega(\log n/\log\log n)$. Therefore, our algorithms are tight within a factor of $O(\log r \log\log n)$ or $O(\log r \log^{\epsilon} n)$.

At the end of this section we also give a general framework that can turn any α approximation algorithm for the RO-AA-MN model, (i.e. the primary variant of the matroid secretary problem) into an $O(\alpha \log^{1+\epsilon} n/\epsilon)$ approximation algorithm in the RO-AA-MU model (see subsection 3.2).

We use the same notation as section 2: $\mathcal{M} = (E, I)$ is a matroid of rank r (which is not known to the algorithm), and $e_1, e_2, \ldots, e_n$ is the the adversarial ordering of the elements of $\mathcal{M}$, and $W = \{w_1 > w_2 > \ldots > w_n\}$ is the set of hidden weights chosen by the adversary that are assigned to the elements of $\mathcal{M}$ via a random bijection $\omega : E \to W$.

3.1 Approximation for AO-RA-MN models

We start by deriving an $O(\log n \log r)$ approximation algorithm for the AO-RA-MN model. Our algorithm basically tries to ignore the the loops and only focuses on the non-loop elements. We design our algorithm in two phases. In the first phase we design a randomized algorithm that works even in the AO-AA-MU model assuming that it has a good estimate on the weight of the largest non-loop element. In particular, fix bijection $\omega : W \to E$, and let e_1^* be the largest non-loop element with respect to ω, and e_2^* be the second largest one. We assume that the algorithm knows a bound $\omega(e_2^*) < L < \omega(e_1^*)$ on the largest non-loop element in advance. We show there is a thresholding algorithm, with a *non-fixed* threshold, that achieves an $O(\log r)$ fraction of the optimum (see subsection 3.1.1).

In order to solve the original problem, in the second phase we divide the non-loop elements into a set of blocks $B_1, B_2, \ldots, B_{\log n}$, and we use the previous algorithm as a module to get an $O(\log r)$ of optimum within each block (see subsection 3.1.2).

3.1.1 Approximation for AO-RA-MN model, with an estimate on the largest weight

Let us start by the first phase. Since our algorithm works in a more general model, here we assume that we are in the AO-AA-MU model, i.e. assuming that both the ordering of the elements and assignments of the weights are chosen adversarially, and the algorithm knows nothing except a bound $\omega(e_2^*) < L < \omega(e_1^*)$ on the largest non-loop element. We design a randomized $O(\log r)$ approximation algorithm for this model.

Note that if r is also known in advance then a simple variant of the thresholding algorithm of Babaioff et al. [3, ThresholdPrice Algorithm] would be a $O(\log r)$ approximation. Indeed it is sufficient to select a threshold $L/2^i$, for $0 \le i \le \log r$ uniformly at random, and then include all the elements above the threshold that are independent of the elements chosen so far. Here, since we do not know r, our algorithm keeps track of the rank of the elements seen so far, and tries to update the threshold according to it. In particular, once the rank of the elements seen so far reaches 2^i, the algorithm inserts a new threshold dynamically and works with it as if it exists since the beginning of the algorithm. The details are described in Algorithm 3:

Let $\mathcal{E}_1$ be the event the algorithm chooses the option in step 1. Also let $r^*(t)$ and $w^*(t)$ be the value of r^* and w^*, respectively, after observing the first t elements of the input. In particular, $r^*(n)$

Algorithm 3 Algorithm for AO-AA-MU model, when an estimate of the largest non-loop element is known

Input: The bound L such that $\omega(e_2^*) < L < \omega(e_1^*)$.
Output: An independent set ALG $\subseteq E$.

with probability 1/2, pick a non-loop element with weight above L and return it.
ALG $\leftarrow \emptyset$ and $r^* \leftarrow 2$.
set threshold $w^* \leftarrow L/2$.
for each arriving element e_i **do**
  if $\omega(e_i) > w^*$ and $ALG \cup \{e_i\}$ is independent **then**
    ALG $\leftarrow$ ALG $\cup \{e_i\}$
  end if
  if $rank(\{e_1, \ldots, e_i\}) \geq r^*$ **then**
    with probability $\frac{1}{\log 2r^*}$ set $w^* \leftarrow L/2r^*$.
    $r^* \leftarrow 2r^*$.
  end if
end for
return ALG

will be the rank of $\mathcal{M}$, and $w^*(n)$ will be the final value of the threshold chosen by the algorithm. The following observation describes some properties of the algorithm:

Observation 3.1. *Assuming $\neg\mathcal{E}_1$, for any matroid of rank r, observe that $r^*(n)$ in the algorithm will be the smallest power of 2 greater than r (i.e. $r^*(n) \leq 2r$). Therefore, the algorithm will choose between at most $\log(2r)$ different thresholds, where for each i, the threshold $w^*(t)$ will be decreased to $L/2^i$ at the first time $t(i)$ where $rank(e_1, \ldots, e_{t(i)}) = 2^{i-1}$, with probability $1/i$.*

Hence, by applying a simple induction it is not hard to see that at any time t in the execution of the algorithm,

$$1 \leq i \leq \log r^*(t), \quad \mathbf{P}\left[w^*(t) = \frac{L}{2^i} \middle| \neg\mathcal{E}_1\right] = 1/\log r^*(t), \tag{4}$$

where the probability is over all of the randomization in the algorithm.

Theorem 3.2. *For any matroid $\mathcal{M} = (E, \mathcal{I})$ of rank r, and any bijection $\omega : E \to W$, given the bound $\omega(e_2^*) < L < \omega(e_1^*)$, Algorithm 3 is a $16 \log r$ approximation in the AO-AA-MU model. i.e.*

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega))\right] \geq \frac{1}{16 \log r} w(\mathrm{OPT}(\omega)),$$

where the expectation is over all of the randomization in the algorithm.

Let us partition the elements of OPT(ω) according to their weights, where

$$1 \leq i \leq \log 2r: \quad P_i = \left\{ e \in \mathrm{OPT}(\omega) : \frac{L}{2^i} < \omega(e) \leq \frac{L}{2^{i-1}} \right\}. \tag{5}$$

First in the next claim, we show that conditioned on $w^*(n) = L/2^i$ (and $\neg\mathcal{E}_1$), the expected weight of ALG(ω), is a constant fraction of $w(P_i)$, unless the size of $|P_i|$ is very small. In the latter case as we will show in equation (7), we may charge $w(P_i)$ by a $1/\log r$ fraction of $\omega(e_1^*)$. Since $\mathcal{E}_1$ occurs with constant probability, the algorithm achieves a constant fraction of $\omega(e_1^*)$ which completes the proof.

Claim 3.3. *For any $1 \le i \le \log r$, if $|P_i| \ge 2^i$, then*

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega))|w^*(n) = \frac{L}{2^i} \wedge \neg\mathcal{E}_1\right] \ge \frac{1}{4}w(P_i).$$

Proof. Let $E_i = \{e_1, \ldots, e_i\}$ be the set of the first i elements. Recall that $t(i)$ is the first time t where $rank(E_t) = 2^{i-1}$. Since $P_i \subseteq \mathrm{OPT}(\omega)$ is an independent set of $\mathcal{M}$, we have $|P_i \cap E_{t(i)}| \le 2^{i-1}$. In other words, we must have seen at most 2^{i-1} elements of the set P_i by the time $t(i)$.

Suppose $w^*(n) = L/2^i$; since $w^*(t)$ is a non-increasing function of t (with probability 1), we get $w^*(t) \ge L/2^i$. Since $P_i \setminus E_{t(i)}$ is an independent set and all its elements will come after $t(i)$, we get $|\mathrm{ALG}(\omega)| \ge |P_i \setminus E_{t(i)}| \ge |P_i| - 2^{i-1}$ by the end of the algorithm. But all these elements are greater than $w^*(n) = L/2^i$, thus:

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega))|w^*(n) = \frac{L}{2^i} \wedge \neg\mathcal{E}_1\right] \ge |P_i \setminus E_{t(i)}|\frac{L}{2^i} \ge \frac{|P_i|L}{2^{i+1}} \ge \frac{1}{4}w(P_i),$$

where the last inequality follows from equation (5). □

Now we are ready to prove Theorem 3.2

Proof of Theorem 3.2. Using the above claim we may simply compute the overall performance of the algorithm:

$$\begin{aligned}
\mathbf{E}\left[w(\mathrm{ALG}(\omega))\right] &= \frac{1}{2}\mathbf{E}\left[w(\mathrm{ALG}(\omega))|\mathcal{E}_1\right] + \frac{1}{2}\mathbf{E}\left[w(\mathrm{ALG}(\omega))|\neg\mathcal{E}_1\right] \\
&\ge \frac{1}{2}\omega(e_1^*) + \frac{1}{2}\sum_{i:|P_i|\ge 2^i} \mathbf{E}\left[w(\mathrm{ALG}(\omega))\middle| w^*(n) = \frac{L}{2^i} \wedge \neg\mathcal{E}_1\right]\mathbf{P}\left[w^*(n) = \frac{L}{2^i}\middle|\neg\mathcal{E}_1\right] \\
&\ge \frac{\omega(e_1^*)}{2} + \frac{1}{2}\sum_{i:|P_i|\ge 2^i} \frac{w(P_i)}{4}\frac{1}{\log 2r} \qquad (6) \\
&\ge \frac{\omega(e_1^*)}{4} + \sum_{i=1}^{\log 2r} \frac{2L}{8\log 2r} + \frac{1}{2}\sum_{i:|P_i|\ge 2^i} \frac{w(P_i)}{4}\frac{1}{\log 2r} \qquad (7) \\
&\ge \frac{\omega(e_1^*)}{4} + \sum_{i=1}^{\log 2r} \frac{w(P_i)}{8\log 2r}, \qquad (8)
\end{aligned}$$

where inequality (6) follows from equation (4) and Claim 3.3, inequality (7) follows from the assumption $\omega(e_1^*) \ge L$, and inequality (8) follows from $w(P_i) \le |P_i|\frac{L}{2^{i-1}} \le 2L$ for $|P_i| \le 2^i$.

The theorem simply follows from the fact that $w(\mathrm{OPT}(\omega)) \le 2(\omega(e_1^*) + \sum w(P_i))$. □

Before describing our algorithm for the AO-RA-MN model, we prove a bound on the performance of algorithm 3 when the bound L can be much larger than the maximum weight (i.e. $\omega(e_1^*) \ll L$). This may happen as a special case when we want to apply Algorithm 3 as a subroutine.

Corollary 3.4. *For any matroid $\mathcal{M} = (E, \mathcal{I})$ of rank r, and any bijection $\omega : E \to W$, given any bound $L > \omega(e_2^*)$ we have*

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega))\right] \ge \max\left(0, \frac{w(\mathrm{OPT}(\omega))}{16\log r} - 2L\right). \qquad (9)$$

If in addition $L < \omega(e_1^*)$*, then*

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega))\right] \geq \frac{\omega(e_1^*)}{2}. \tag{10}$$

Proof. To prove the first inequality, note that if $L < \omega(e_1^*)$, then we are done, otherwise suppose that we increase the weight of e_1^* to $L + \omega(e_1^*)$. Define $\omega' = \omega$ on all elements, except $\omega'(e_1^*) = L + \omega(e_1^*) \leq 2L$. Then by Theorem 3.2, we have

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega'))\right] \geq \frac{w(\mathrm{OPT}(\omega'))}{16 \log r} = \frac{w(\mathrm{OPT}(\omega)) + L}{16 \log r}$$

On the other hand, since in the worst case $\mathrm{ALG}(\omega)$ does not have e_1^*, while $\mathrm{ALG}(\omega')$ has it, we have $\mathbf{E}\left[w(\mathrm{ALG}(\omega))\right] \geq \mathbf{E}\left[w(\mathrm{ALG}(\omega'))\right] - 2L$. Therefore

$$\mathbf{E}\left[w(\mathrm{ALG}(\omega))\right] \geq \frac{w(\mathrm{OPT}(\omega)) + L}{16 \log r} - 2L \geq \max\left(0, \frac{w(\mathrm{OPT}(\omega))}{16 \log r} - 2L\right).$$

The second inequality can be proved simply by noting that the algorithm picks e_1^* in step 1 with probability $1/2$. □

3.1.2 Approximation for AO-RA-MN by a general reduction

Now we are ready to describe our final algorithm for AO-RA-MN model without knowing L in advance (here, unlike the previous algorithm we will use the random assignment of weights). The idea is to only consider the non-loop elements and divide them into a set of blocks $B_1, B_2, \ldots, B_{\log 2n}$ such that $|B_i| = 2^i$ (note that the number of non-loop elements can be quite smaller than n, but we do not know it in advance). After observing the first i blocks, we would have a good guess on the largest weight of the next block. Using that guess as a bound L, with probability $1/\log(2n)$, we run Algorithm 3 on block $i+1$ and return its solution as the final answer. The details are described in Algorithm 4.

Algorithm 4 Algorithm for AO-RA-MN model

Input: n, the number of elements.

Output: An independent set $\mathrm{ALG} \subseteq E$.

1: Choose a number $0 \leq b \leq \log n$ uniformly at random.
2: Observe the first $2^b - 1$ non-loop elements without picking any of them, and let $L(b)$ be the largest weight among these non-loop elements.
3: Run Algorithm 3 only on the next 2^b non-loop elements (ignore loops), with parameters $n = 2^b$ and $L = L(b)$, and return its output.

The next theorem proves the correctness of the algorithm

Theorem 3.5. *For any matroid* $\mathcal{M} = (E, \mathcal{I})$ *of rank* r*, Algorithm 4 is a* $O(\log r \log n)$ *approximation in the AO-RA-MN model.*

Let F be the set of non-loop elements, $m := |F|$, and let $F_i \subset F$ be the set of first $2^{i+1} - 1$ non-loop elements (as a special case $F_{\log m} = F$. We divide the elements of F into a set of blocks

$B_0, B_1, \ldots, B_{\lfloor \log m \rfloor}$, where $B_0 := F_0$, and for each $i > 0$, $B_i := F_i \setminus F_{i-1}$. Note that the size of the last block $|B_{\log m}| = m + 1 - 2^{\lfloor \log m \rfloor}$ can be much smaller than $2^{\lfloor \log m \rfloor}$.

For a set of weights $W' \subset W$ and $E' \subset E$ of elements such that $|W'| = |E'|$, let $\mathcal{E}_{W'}(E')$ be the event $\omega(E') = W'$). Fix a set $W' \subset W$ of size $|W'| = |F|$. Throughout the proof we always condition on $\mathcal{E}_{W'}(F)$. Define

$$0 \le i \le \lfloor \log m \rfloor : \quad O_i = \mathbf{E}_\omega \left[w(\mathrm{OPT}(\omega) \cap B_i) | \mathcal{E}_{W'}(F) \right], \tag{11}$$

to be the expected value of the optimum set in each of the blocks. We will show that

$$\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) | \mathcal{E}_{W'}(F) \right] \ge \frac{1}{2500 \log r \log n} \sum_{i=1}^{\log m} B_i.$$

In the next claim we show that conditioned on algorithm chooses $b = i$ in the step 1, it will get an $\Omega(1/\log r)$ fraction of O_i. Note that in this claim we do not analyze the special case of $b = \lfloor \log m \rfloor$.

Claim 3.6. *If the algorithm chooses $b = i < \lfloor \log m \rfloor$ in step 1, it will get an $\Omega(1/\log r)$ fraction of O_i:*

$$\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) | b = i, \mathcal{E}_{W'}(F) \right] \ge \frac{1}{128 \log r} O_i.$$

Proof. Fix a set of weights $S = \{s_1 > s_2 > \ldots > s_{2^{i+1}-1}\} \subset W'$. Conditioned on $\mathcal{E}_S(F_i)$, there is a constant probability that $s_1 \in \omega(B_i)$ and $s_2 \notin \omega(B_i)$; thus $L(b) = s_2$ will be a feasible bound for Algorithm 3. Therefore, we may apply Theorem 3.2 and obtain $\Omega(\log r)$ fraction of O_i. Thus

$$\begin{aligned}
&\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) | \mathcal{E}_S(F_i), b = i, \mathcal{E}_{W'}(F) \right] \ge \\
&\quad \ge \frac{1}{4} \mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) | s_1 \in B_i, s_2 \notin B_i, \mathcal{E}_S(F_i), b = i, \mathcal{E}_{W'}(F) \right] \\
&\quad \ge \frac{1}{64 \log r} \mathbf{E}_\omega \left[w(\mathrm{OPT}(\omega) \cap B_i) | s_1 \in B_i, s_2 \notin B_i, \mathcal{E}_S(F_i), \mathcal{E}_{W'}(F) \right] \qquad (12) \\
&\quad \ge \frac{1}{128 \log r} \mathbf{E}_\omega \left[w(\mathrm{OPT}(\omega) \cap B_i) | \mathcal{E}_S(F_i), \mathcal{E}_{W'}(F) \right]. \qquad (13)
\end{aligned}$$

Here inequality (12) follows from Theorem 3.2, and inequality (13) holds by noting that removing the condition $s_2 \notin B_i$ can only double the expectation of OPT, while removing $s_1 \notin B_i$ may only decrease its expectation. The claim simply follows by summing up inequality (13) over all events $\mathcal{E}_S(F_i)$, for any $S \subset W', |S| = 2^{i+1} - 1$.

□

Now we are ready to Prove Theorem 3.5

Proof of Theorem 3.5. We use Claim 3.6 to lower bound the expected gain of the algorithm from all except the last block. We need to analyze $b = \lfloor \log m \rfloor$ differently. Indeed if $B_{\lfloor \log m \rfloor} \ll m/2$, the bound $L(b)$ will be much larger than the largest weight in $\omega(B_{\lfloor \log m \rfloor})$ w.h.p. Therefore, we apply Corollary 3.4 for this special case. Intuitively, the loss incurs by misreporting the bound $L(\lfloor \log m \rfloor)$ is no more than the largest weight in W', and this can be compensated simply by selecting the largest weight with constant probability.

Let L' be the largest weight in W'. By Corollary 3.4 (equation (9)), we obtain

$$\mathbf{E}_\omega \left[w(\mathrm{ALG}(\omega)) | b = \lfloor \log m \rfloor, \mathcal{E}_{W'}(F) \right] \ge \max \left(0, \frac{O_{\lfloor \log m \rfloor}}{16 \log r} - 2L' \right).$$

Therefore, by Claim 3.6 and the above inequality we get:

$$\begin{aligned}\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))\right] &= \sum_{i=0}^{\lfloor \log m \rfloor} \mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))|b=i,\mathcal{E}_{W'}(F)\right]\mathbf{P}_\omega\left[b=i|\mathcal{E}_{W'}(F)\right] \\ &\geq \frac{1}{\log 2n}\left(\sum_{i=0}^{\lfloor \log m \rfloor -1}\frac{O_i}{128\log r}+\max\{0,\frac{O_{\lfloor \log m \rfloor}}{16\log r}-2L'\}\right). \qquad (14)\end{aligned}$$

In order to lower bound the RHS it suffices to show that $\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))|\mathcal{E}_{W'}\right]=\Omega(L'/\log n)$. This simply follows from the second part of Corollary 3.4. For any block B_i, conditioned on $L'\in\omega(B_i)$, with probability $1/2$, the second largest weight in $\omega(F_i)$, is not assigned to B_i, in which case algorithm achieves L' with probability $1/2$, once it chooses $b=i$:

$$\begin{aligned}\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))|\mathcal{E}_{W'}(F)\right] &= \sum_{i=0}^{\log m}\frac{|B_i|}{\log m}\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))|L'\in\omega(B_i),\mathcal{E}_{W'}(F)\right] \\ &= \sum_{i=0}^{\log m}\frac{|B_i|}{\log m}\frac{\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))|b=i,L'\in\omega(B_i),\mathcal{E}_{W'}(F)\right]}{\log 2n} \\ &\geq \sum_{i=0}^{\log m}\frac{|B_i|L'}{4\log m\log 2n}=\frac{L'}{4\log 2n}. \qquad (15)\end{aligned}$$

Therefore, by adding up equation (14) and 8 times equation (15) we obtain

$$\begin{aligned}9\mathbf{E}_\omega\left[w(\mathrm{ALG}(\omega))|\mathcal{E}_{W'}\right] &\geq \frac{1}{\log 2n}\left(\sum_{i=0}^{\lfloor \log m \rfloor -1}\frac{O_i}{128\log r}+\max\{0,\frac{O_{\lfloor \log m \rfloor}}{16\log r}-2L'\}+2L'\right) \\ &\geq \sum_{i=0}^{\lfloor \log m \rfloor}\frac{O_i}{128\log r\log 2n}=\Omega\left(\frac{1}{\log r\log n}\right)\mathbf{E}_\omega\left[w(\mathrm{OPT}(\omega))|\mathcal{E}_{W'}\right].\end{aligned}$$

Summing both sides of the inequality over all events $\mathcal{E}_{W'}$ completes the proof. □

3.2 Matroid secretary with unknown n

In this subsection we consider the primary variant of the matroid secretary problem.When the total number of elements n is known in advance (RO-AA-MN model), there is an $O(\log r)$-approximation which was designed in [3] and is still the best known approximation for this problem.

Here we show a simple reduction which implies that if we do not have any information about the matroid or the number of elements (the RO-AA-MU model), we can achieve an $O(\frac{1}{\epsilon}\log^{1+\epsilon} n\log r)$-approximation for any fixed $\epsilon>0$.

Theorem 3.7. *Let $\mathcal{M}$ be a matroid of rank r on n elements. If there is an α approximation algorithm for the matroid secretary problem on $\mathcal{M}$ in the RO-AA-MN model, then for any fixed $\epsilon>0$, there is also an $O(\frac{\alpha}{\epsilon}\log^{1+\epsilon} n)$-approximation for the matroid secretary problem on $\mathcal{M}$ with no information given in advance (the RO-AA-MU model).*

Proof. We guess a number n' according to a probability distribution with a polynomial tail, as follows: let $n' = 2^i$ where $i \geq 0$ is chosen with probability

$$p_i = \frac{\epsilon}{1+\epsilon} \cdot \frac{1}{(1+i)^{1+\epsilon}}.$$

This distribution is chosen so that $\sum_{i=1}^{\infty} p_i \leq 1$ (with the remaining probability, we do nothing); this can be verified as follows:

$$\sum_{i=0}^{\infty} \frac{1}{(1+i)^{1+\epsilon}} = 1 + \sum_{i=1}^{\infty} \frac{1}{(1+i)^{1+\epsilon}} \leq 1 + \int_0^{\infty} \frac{dx}{(1+x)^{1+\epsilon}} = 1 + \left[-\frac{1/\epsilon}{(1+x)^{\epsilon}} \right]_0^{\infty} = 1 + \frac{1}{\epsilon}.$$

Then we run the α-approximation algorithm as a black box, under the assumption that the number of elements is n'.

Assume that the actual number of elements is $n \in [2^i, 2^{i+1})$. With probability p_i, our guess of the number of elements is $n' = 2^i$. If this happens, we retrieve $1/\alpha$ of the expected value of the optimal solution on the first n' elements. Since the elements arrive in a random order, the expected optimum on the first n' elements is at least $1/2$ of the actual optimum. Hence, in expectation we obtain at least

$$p_i \frac{OPT}{2\alpha} \geq \frac{\epsilon}{1+\epsilon} \cdot \frac{1}{(1+i)^{1+\epsilon}} \cdot \frac{OPT}{2\alpha} \geq \frac{\epsilon}{4\alpha} \frac{1}{(1+\log n)^{1+\epsilon}} OPT.$$

□

Therefore, if we run the $O(\log r)$ approximation of Babaioff et al. [3] as a black box we achieve an $O(\frac{1}{\epsilon} \log^{1+\epsilon} n \log r)$ for the RO-AA-MN model:

Corollary 3.8. *For any fixed $\epsilon > 0$, there is an $O(\frac{1}{\epsilon} \log^{1+\epsilon} n \log r)$-approximation for the matroid secretary problem for a matroid $\mathcal{M}$ of rank r on n elements, with no information given in advance (the RO-AA-MU model). In particular, assuming that $\mathcal{M}$ is a partition matroid matroid of rank 1, we obtain an $O(\log^{1+\epsilon} n/\epsilon)$ approximation for the classical secretary problem, with no information given in advance(the CU model).*

We shall see in Section 4.2 that even in the case of $r = 1$ (expectation-maximizing classical secretary problem) where n is chosen adversarially from $\{1, \ldots, N\}$, we cannot achieve a factor better than $O(\log N / \log \log N)$.

4 Classical secretary with unknown n

In this section, we consider a variant of the classical secretary problem where we want to select exactly one element (i.e. in matroid language, we consider a uniform matroid of rank 1). However, here we assume that the total number of elements n (which is crucial in the classical $1/e$-competitive algorithm) is not known in advance - it is chosen by an adversary who can effectively terminate the input at any point. We consider the worst case, i.e. we want to achieve a certain probability of success regardless of when the input is terminated. We show that there is no algorithm achieving a constant probability of success in this case. However, we can achieve logarithmic guarantees and also prove closely matching lower bounds (see subsection 4.1).

In subsection 4.2 we show that even if we want to maximize the expected weight of the selected element, and n is known to be upper bounded by N, still no algorithm can achieve a better than $\Omega(\log N/\log\log N)$ approximation factor in expectation. Consequently, we obtain that no algorithm can achieve an approximation factor better than $\Omega(\log N/\log\log N)$ in the AO-RA-MN model.

4.1 Known upper bound on n

First, let us consider the following scenario: an upper bound N is given such that the actual number of elements on the input is guaranteed to be $n \in \{1, 2, \ldots, N\}$. The adversary can choose any n in this range and we do not learn n until we process the n-th element. (e.g., we are interviewing candidates for a position and we know that the total number of candidates is certainly not going to be more than 1000. But, we might run out of candidates at any point.) The goal is to select the highest-ranking element with a certain probability. Assuming the *comparison model* (i.e., where only the relative ranks of elements are known to the algorithm), we show that there is no algorithm achieving a constant probability of success in this case.

Theorem 4.1. *Given that the number of elements is chosen by an adversary in $\{1, \ldots, N\}$ and N is given in advance, there is a randomized algorithm which selects the best element out of the first n with probability at least $1/(H_{N-1}+1)$.*

On the other hand, there is no algorithm in this setting which returns the best element with probability more than $1/H_N$. Here, $H_N = \sum_{i=1}^{n} \frac{1}{i}$ is the N-th harmonic number.

Our proof is based on the method of Buchbinder et al. [6] which bounds the optimal achievable probability by a linear program. In fact the optimum of the linear program is *exactly* the optimal probability that can be achieved.

Lemma 4.2. *Given the classical secretary problem where the number of elements is chosen by an adversary from $\{1, 2, \ldots, N\}$ and N is known in advance, the best possible probability with which an algorithm can find the optimal element is given by*

$$\begin{aligned} \max \quad & \alpha : \\ \forall n \le N; \quad & \tfrac{1}{n}\textstyle\sum_{i=1}^{n} i p_i \ge \alpha, && (16) \\ \forall i \le N; \quad & \textstyle\sum_{j=1}^{i-1} p_j + i p_i \le 1, && (17) \\ \forall i \le N; \quad & p_i \ge 0. \end{aligned}$$

The only difference between this LP and the one in [6] is that we have multiple constraints (16) instead of what is the objective function in [6]. We use essentially the same proof to argue that this LP captures *exactly* the optimal probability of success α that an algorithm can achieve. We give the proof for completeness; understanding the validity of this LP will be also useful for us later.

Proof. Consider any (randomized) algorithm which finds the best element with probability at least α, for every possible number of incoming elements $n \in \{1, \ldots, N\}$. It is convenient to assume that the algorithm never learns n and possibly continues running beyond the first n elements (in which case it has failed). Let us define

$$p_i = \mathbf{P}\left[\text{algorithm skips the first } i-1 \text{ candidates and chooses candidate } i\right].$$

The probability here is over both the randomness on the input and the randomness of the algorithm itself. Recall that the actual number of candidates n is not known beforehand. All that the

algorithm knows at time i are the relative ranks of the first i candidates, which are also independent of n. So the probabilities p_i cannot depend on n.

Note that these are probabilities of disjoint events, so we have $\sum_{i=1}^{n} p_i \leq 1$. The LP actually contains stronger inequalities (17). The reason why these inequalities are valid is as follows: We can assume w.l.o.g. that the algorithm never selects an element which is not the best so far. (Any algorithm can be converted to this form and perform at least as well.) The probability (over random permutations of the input) that the i-th candidate is the best so far is $1/i$. Therefore,

$$\mathbf{P}\left[\text{algorithm skips the first } i-1 \text{ and chooses } i \mid \text{candidate } i \text{ is the best out of the first } i\right] =$$

$$= \frac{\mathbf{P}\left[\text{algorithm skips the first } i-1 \text{ and chooses candidate } i\right]}{\mathbf{P}\left[\text{candidate } i \text{ is the best out of the first } i\right]} = ip_i.$$

On the other hand, the probability that the algorithm skips the first $i-1$ elements is $1 - \sum_{j=1}^{i-1} p_j$. This event is independent of whether the i-th element is the best among the first i, because all the algorithm learns about the first $i-1$ elements are their relative ranks. This proves the constraint (17):

$$1 - \sum_{j=1}^{i-1} p_j = \mathbf{P}\left[\text{algorithm skips the first } i-1 \mid \text{candidate } i \text{ is the best among the first } i\right] \leq ip_i.$$

The probability that the i-th candidate is the actual best candidate among the first n is $1/n$. Conditioned on this event, candidate i is also the best among the first i candidates (and that is the only information available to the algorithm at that moment), so the algorithm selects candidate i with conditional probability exactly ip_i. The total probability that the algorithm selects the best candidate out of the first n elements is

$$\mathbf{P}\left[\text{success}\right] = \sum_{i=1}^{n} \mathbf{P}\left[\text{element } i \text{ is optimal } \& \text{ algorithm selects } i\right] = \sum_{i=1}^{n} \frac{1}{n} \cdot ip_i.$$

We assume that the algorithm achieves success probability α for any number of candidates $n \in \{1, \ldots, N\}$ chosen by an adversary. This proves the constraint (16).

Conversely, given a feasible solution to this LP, an algorithm can proceed as follows (see [6]): If it comes to the i-th element and this is the best element so far, take it with probability $ip_i/(1 - \sum_{j=1}^{i-1} p_j)$ (which is at most 1 by (17). It can be verified by induction that the probability of skipping the first $i-1$ elements and finding that element i is the best so far is $(1 - \sum_{j=1}^{i} p_j)/i$, and hence the total probability of taking element i is exactly p_i. Conditioned on element i being the actual optimum (which happens with probability $1/n$), we take it with probability ip_i. By (16), the success probability is at least α for any input length n. □

For a given N, an algorithm can explicitly solve the LP given by Lemma 4.2 and thus achieve the optimal probability. Theorem 4.1 can be proved by estimating the value of this LP.

Proof of Theorem 4.1. First, we show a feasible solution with $\alpha = \frac{1}{H_{N-1}+1}$. We define $p_i = \frac{1}{i(H_{N-1}+1)}$ for each $i = 1, \ldots, N$. This induces an algorithm as described above: if it comes to the i-the element and it is the best so far, we take it with probability

$$\frac{ip_i}{1 - \sum_{j=1}^{i-1} p_j} = \frac{1}{H_{N-1} + 1 - H_{i-1}}.$$

By Lemma 4.2, it is sufficient to verify that (p_i, α) is a feasible solution:

$$\frac{1}{n}\sum_{i=1}^{n} ip_i = \frac{1}{H_{N-1}+1} = \alpha$$

implies (16), and

$$ip_i + \sum_{j=1}^{i-1} p_j = \frac{1}{H_{N-1}+1}(1 + \sum_{j=1}^{i-1}\frac{1}{j}) = \frac{1}{H_{N-1}+1}(1 + H_{i-1}) \leq 1$$

implies (17). This proves that there is an algorithm with probability of success $1/(H_{N-1}+1)$.

Conversely, we prove that for any feasible solution, we have $\alpha \leq 1/H_N$. For this, we in fact consider a weaker LP:

$$\begin{aligned} & \max && \alpha : \\ \forall n \leq N; \quad & && \tfrac{1}{n}\textstyle\sum_{i=1}^{n} ip_i \geq \alpha, && (18) \\ & && \textstyle\sum_{i=1}^{N} p_i \leq 1, && (19) \\ \forall i \leq N; \quad & && p_i \geq 0. \end{aligned}$$

Obviously, any feasible solution to (16-17) is also feasible for (18-19). Fixing α, consider a feasible solution to (18-19) which minimizes $\sum_{i=1}^{N} p_i$. We claim that $ip_i \geq \alpha$ for each i. If not, take the first index j such that $jp_j < \alpha$. By (18) for $n = j$, there must be a smaller index $j' < j$ such that $j'p_{j'} > \alpha$. Then we can decrease $p_{j'}$ by δ/j' and increase p_j by δ/j for some small $\delta > 0$, so that $j'p_{j'} + jp_j$ is preserved. We can make sure that no inequality (18) is violated, because the left-hand side is preserved for all $n \geq j$, and the inequality was not tight for $j' \leq n < j$. On the other hand, $\sum_{i=1}^{N} p_i$ decreases by $\delta/j' - \delta/j$. This is a contradiction.

Therefore, we have $p_i \geq \alpha/i$ for all i. By summing up over all i and using $\sum_{i=1}^{N} p_i \leq 1$, we get

$$1 \geq \sum_{i=1}^{N} p_i \geq \alpha \sum_{i=1}^{N} \frac{1}{i} = \alpha H_N.$$

□

4.2 Maximizing the expected weight

A slightly different model arises when elements arrive with (random) weights and we want to maximize the expected weight of the selected element. This model is somewhat easier for an algorithm; any algorithm that selects the best element with probability at least α certainly achieves an α-approximation in this model, but not the other way around. Given an upper bound N on the number of elements (and under a more stringent assumption that weights are chosen i.i.d. from a known distribution), by a careful choice of a probability distribution for the weights, we prove that still no algorithm can achieve an approximation factor better than an $\Omega(\log N/\log\log N)$-approximation.

Theorem 4.3. *For the classical secretary problem with random nonnegative weights drawn i.i.d. from a known distribution and the number of candidates chosen adversarially in the range $\{1, \ldots, N\}$, no algorithm achieves a better than $\frac{\log N}{32 \log\log N}$-approximation in expectation.*

The hard examples are constructed based on a particular exponentially distributed probability distribution. Similar constructions have been used in related contexts [16, 10]. *Proof.* We define a probability distribution over weights as follows. For a parameter $\gamma \in (0, \frac{1}{3})$ (possibly depending on N), let the weight of each element be (independently)

- $w_j = 2^{\gamma j}$ with probability $1/2^j$, for each $j \geq 1$.

Note that although the weights are unbounded, the expected weight of each element is finite.

Consider blocks of elements where the i-th block B_i has size 2^i. The adversary will choose arbitrarily a number of blocks $\ell \leq \log N$, and a stopping point $n = \sum_{i=1}^{\ell} 2^i = 2^{\ell+1} - 1$. Note that given ℓ, the expected optimum is

$$OPT_\ell \geq \sum_{j=1}^{\infty} w_j \mathbf{P}\left[w_j \text{ is the largest weight among } 2^\ell \text{ elements}\right].$$

The probability that no weight larger than w_j appears among 2^ℓ elements is $(1 - 1/2^j)^{2^\ell}$. So,

$$\begin{aligned}\mathbf{P}\left[w_j \text{ is the largest weight among } 2^\ell \text{ elements}\right] &= (1 - 1/2^j)^{2^\ell} - (1 - 1/2^{j-1})^{2^\ell} \\ &= \Theta(\min\{2^{\ell-j}, 1\}).\end{aligned}$$

Therefore, since $w_j = 2^{\gamma j}$ and $\gamma \in (0, \frac{1}{3})$, the expected contribution from elements of weight w_j is roughly $2^{\gamma j}\min\{2^{\ell-j}, 1\}$, which is maximized for $j = \ell$. (Note also that the distribution decays exponentially both for $j > \ell$ and $j < \ell$.) So the largest contribution comes from elements of weight roughly w_ℓ. We can estimate:

$$OPT_\ell \geq w_\ell \mathbf{P}\left[w_\ell \text{ appears among } 2^\ell \text{ elements}\right] = 2^{\gamma\ell}(1 - (1 - 1/2^\ell)^{2^\ell}) \geq (1 - 1/e)2^{\gamma\ell}.$$

Now consider any algorithm (which does not know ℓ beforehand). Let p_i denote the probability that the algorithm skips the first $i-1$ blocks and then chooses some element in block B_i. Note that this event might be correlated with the random weights that appear in blocks $B_1, \ldots, B_i$. However, we have a bound on the probability that weight w_j appears in block B_i:

$$\mathbf{P}\left[w_j \text{ appears in block } B_i\right] = 1 - (1 - 1/2^j)^{2^i} \leq \min\{2^{i-j}, 1\}.$$

Let p_{ij} denote the probability that the algorithm gets an element of weight w_j from block B_i. By the above we have $p_{ij} \leq \min\{2^{i-j}, 1\}$. Also, by definition of the probabilities, $\sum_{j=1}^{\infty} p_{ij} = p_i$. Given p_{ij}, the expected weight that the algorithm obtains from block B_i is $\mathbf{E}\left[\text{profit from } B_i\right] = \sum_{j=1}^{\infty} w_j p_{ij}$ and the total profit over the first ℓ blocks is $\sum_{i=1}^{\ell}\sum_{j=1}^{\infty} w_j p_{ij}$. Thus the expected profit of any algorithm can be bounded by the following LP.

$$\begin{aligned}\max \quad & \alpha : \\ \forall \ell \leq \log N; \quad & \textstyle\sum_{i=1}^{\ell}\sum_{j=1}^{\infty} w_j p_{ij} \geq \alpha OPT_\ell; \\ \forall i, j; \quad & p_{ij} \leq \min\{2^{i-j}, 1\}; \\ \forall i; \quad & \textstyle\sum_{j=1}^{\infty} p_{ij} = p_i; \\ & \textstyle\sum_{i=1}^{\ell} p_i \leq 1; \\ & p_i \geq 0.\end{aligned}$$

We estimate the value of this LP as follows. Subject to the condition $\sum_{j=1}^{\infty} p_{ij} = p_i$, the quantity $\sum_{j=1}^{\infty} w_j p_{ij}$ will be maximized if we make p_{ij} for large j as large as possible. However, note that $2^{\gamma j} p_{ij} \le 2^{\gamma j} 2^{i-j}$, so the tail for $j \to \infty$ decays exponentially and we might as well concentrate only on the first term. Assuming that $p_i = 2^{i-k}$, the best choice is to set $p_{ij} = 0$ for $j \le k$ and $p_{ij} = 2^{i-j}$ for all $j \ge k+1$, which gives

$$\sum_{j=1}^{\infty} w_j p_{ij} \le \sum_{j=k+1}^{\infty} 2^{\gamma j} 2^{i-j} \le \frac{1}{1-2^{\gamma-1}} 2^{(\gamma-1)(k+1)+i} \le 2 \cdot (2^{i-k})^{1-\gamma} 2^{\gamma i}$$

where we used $\gamma \in (0, \frac{1}{3})$. Note that for any value of p_i, we can apply this argument to the power of 2 nearest to p_i; hence,

$$\sum_{j=1}^{\infty} w_j p_{ij} \le 4 \cdot p_i^{1-\gamma} 2^{\gamma i}.$$

Now suppose the adversary stops the game after ℓ blocks. The expected optimum is $OPT_\ell \ge (1-1/e)2^{\gamma \ell}$ (see above), while the algorithm gets

$$\sum_{i=1}^{\ell} \sum_{j=1}^{\infty} w_j p_{ij} \le 4 \cdot \sum_{i=1}^{\ell} p_i^{1-\gamma} 2^{\gamma i}.$$

This should be at least $\alpha OPT_\ell \ge \alpha(1-1/e)2^{\gamma \ell}$; therefore, we get

$$\sum_{i=1}^{\ell} p_i^{1-\gamma} 2^{\gamma(i-\ell)} \ge \frac{1}{4}(1-1/e)\alpha \ge \frac{1}{8}\alpha.$$

We sum up these inequalities for $\ell = 1, \ldots, \log N$:

$$\sum_{\ell=1}^{\log N} \sum_{i=1}^{\ell} p_i^{1-\gamma} 2^{\gamma(i-\ell)} = \sum_{i=1}^{\log N} p_i^{1-\gamma} \sum_{\ell=i}^{\log N} 2^{\gamma(i-\ell)} \ge \frac{1}{8}\alpha \log N.$$

The sum $\sum_{\ell=i}^{\log N} 2^{\gamma(i-\ell)}$ is bounded by $\sum_{\ell=i}^{\infty} 2^{\gamma(i-\ell)} = \frac{1}{1-2^{-\gamma}} \le \frac{2}{\gamma}$. Therefore, we get

$$\alpha \le \frac{16}{\gamma \log N} \sum_{i=1}^{\log N} p_i^{1-\gamma}.$$

Given that $\sum_{i=1}^{\log N} p_i = 1$ and the function $x^{1-\gamma}$ is concave, the best value of α can be achieved if we set $p_i = 1/\log N$ for all i. Then, we have

$$\alpha \le \frac{16}{\gamma (\log N)^{1-\gamma}}.$$

Finally, we set $\gamma = 1/\log\log N$ which gives

$$\alpha \le \frac{32 \log\log N}{\log N}.$$

□

Consequently, we obtain that no algorithm can achieve an approximation factor better than $\Omega(\log N / \log\log N)$ in the AO-RA-MN model.

Corollary 4.4. *For the matroid secretary problem in the AO-RA-MN (and AO-RA-MU, RO-AA-MU) models, no algorithm can achieve a better than* $\Omega(\frac{\log N}{\log \log N})$*-approximation in expectation.*

Proof. It is not hard to convert the example of Theorem 4.3 into a hard example for the AO-RA-MN model. It suffices to let $\mathcal{M}$ to be a partition matroid of rank 1, and let the first n elements of the inputs to be non-loop while the rest of the input contains only loops. Since the algorithm does not know n in advance (it only knows $n \leq N$), it essentially has to choose one of the first n elements without knowing n, which is a secretary problem where the number of candidates is chosen adversarially in the range $\{1, \ldots, N\}$. Therefore, no algorithm can achieve an approximation factor better than $\Omega(\log N / \log \log N)$ (the same is also true for the AO-RA-MU, RO-AA-MU model, where nothing is known about n in advance). □

5 Conclusion and open questions

We presented a number of positive and negative results for variants of the matroid secretary problem. The main open question is if there is a constant-factor approximation in the RO-AA-MN model, where weights are assigned to elements adversarially and the input ordering of elements is random. An easier question might be whether this is possible in the RO-RA-MN model where both the input order and weight assignment are random, but only the total number of elements n is known in advance (as opposed to the full matroid structure, as in [21]). Note that under an adversarial assignment of weights, knowing the matroid beforehand (RO-AA-MK) does not seem to be easier than the RO-AA-MN model; the true input could be embedded in a much larger matroid with most weights set to zero. A similar question arises for the AO-RA-MN model: whether it is possible to improve the $O(\log n \log r)$ factor, thus closing the gap with the lower-bound of $\Omega(\log n / \log \log n)$.

References

[1] A. R. Abdel-Hamid, J.A. Bather and G. B. Trustrum. The secretary problem with an unknown number of candidates. *J. Appl. Prob.* 19, 619–630, 1982.

[2] M. Babaioff, M. Dinitz, A. Gupta, N. Immorlica and K. Talwar. Secretary problems: weights and discounts. In *SODA 2009*, 1245–1254.

[3] M. Babaioff, N. Immorlica and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA 2007*, 434–443.

[4] M. H. Bateni, M. T. Hajiaghayi and M. Zadimoghaddam. Submodular secretary problem and extensions. In *APPROX 2010*, 39–52.

[5] P. Borosan and M. Shabbir. A survey of secretary problem and its extensions. unpublished manuscript, 2009, available at http://paul.rutgers.edu/~mudassir/Secretary/paper.pdf.

[6] N. Buchbinder, K. Jain and M. Singh. Secretary problems via linear programming. In *IPCO 2010*, 163–176.

[7] S. Chakraborty and O. Lachish. Improved competitive ratio for the matroid secretary problem. To appear in *SODA 2012*.

[8] N. B. Dimitrov and C. G. Plaxton. Competitive weighted matching in transversal matroids. In *ICALP 2008*, 397–408.

[9] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Soviet Mathematics, Doklady* 4, 1963.

[10] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni and C. Stein. Online stochastic packing applied to display ad allocation. In *ESA 2010*, 182–194.

[11] T.S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4:3, 282–289, 1989.

[12] M. Gardner. Mathematical Games column, *Scientific American*, February 1960.

[13] J. Gilbert and F. Mosteller. Recognizing the maximum of a sequence. *J. Amer. Statist. Assoc.* 61:35–73, 1966.

[14] A. Gupta, A. Roth, G. Schoenebeck and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE 2010*, 246–257.

[15] M. T. Hajiaghayi, R. Kleinberg and D. Parkes. Adaptive limited-supply online auctions. In *EC 2004*, 71–80.

[16] M. T. Hajiaghayi, R. Kleinberg and T. Sandholm. Automated online mechanism design and prophet inequalities. In *International Conference on Artificial Intelligence 2007*, 58–65, 2007.

[17] S. Im and Y. Wang. Secretary problems: Laminar matroid and interval scheduling. In *SODA 2011*, 1265-1274.

[18] D. V. Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 10:1, 39–51, 1961.

[19] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA 2005*, 630–631.

[20] N. Korula and M. Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP 2009*, 508–520.

[21] J. A. Soto. Matroid secretary problem in the random assignment model. In *SODA 2011*, 1275-1284.

[22] J. Soto. Contributions on secretary problems, independent sets of rectangles and related problems. PhD Thesis, Department of Mathematics, Massachusetts Institute of Technology, 2011.

[23] T. J. Stewart. The secretary problem with an unknown number of options. *Operations Research* 29:1, 130–145, 1981.